TOPIC G: Integration Testing

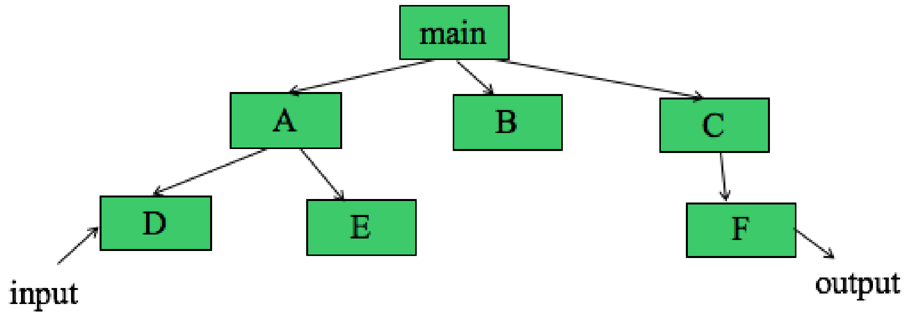
Ch. 6.1 and 6.2

Outline

- Big bang
- Top-down
- Bottom-up
- Sandwich
- Regression
- Two-tier and multi-tier integration testing
- Other types of integration testing

Integration Testing

- Assumes the units have been tested individually
- Tests units working together
- Identifies errors in the interfaces between units
- The goal is to ensure that components work fine when assembled
- Approaches
  - Big bang
  - Top-down
  - Bottom-up
  - Sandwich



Integration Testing – Possible Issues

- Assuming that units/components are tested, integration issues may arise due to Interfacing
    - Procedure/Method calls
    - Shared Memory
    - Message passing
    - Wrong assumption about provided functionality
    - Wring assumptions about the interface (wrong parameters, precondition checks)
    - Wrong error processing
    - Wrong assumption about events timing

Integration testing Stubs

- Stubs replace modules
  - Stub for input: the Stub produces test input data
  - Stub for output: returns test results
- Stubs can replace the whole component
  - For example, network, or a resource
- Stubs must be declared and invoked as the real module
  - Same name
  - Same parameter list
  - Same return type

Example Testing Stub

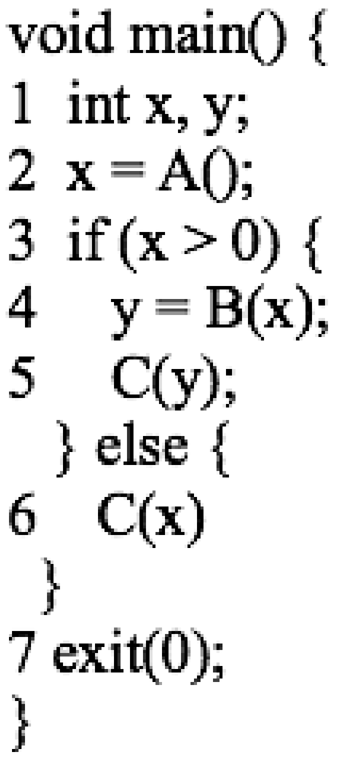
public static int someFunc(float fThat)

```
public static int someFunc(float fThat) {
  int iDummy = 42;
  System.out.println (“In method someFunc ” +
      “Input float =” + fThat);
  System.out.println (“Returning 42.”);
  return iDummy;
}
```

Integration testing Stubs

- Common functions of a stub
  - display/log trace message
  - display/log passed parameters
  - Return value according to test objective
    - from a table
    - from an external file
    - based on a search according to parameter value
    - ..

Example



```
void main() {
1  int x, y;
2  x = A();
3  if (x > 0) {
4      y = B(x);
5      C(y);
   } else {
6    C(x)
  }
7 exit(0);
}
```

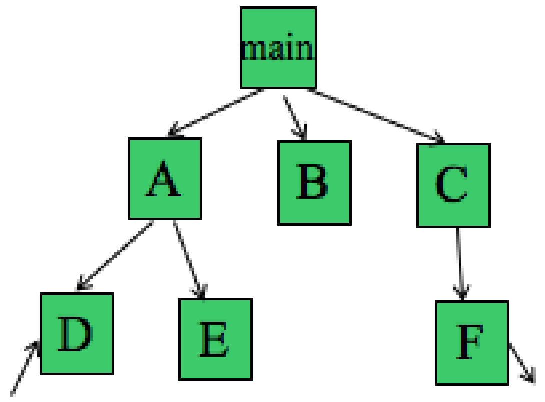
- If you want to test the Path 1-2-3-4-5-7
  - Use a stub for A() such that x>0 is returned
- If you want to test for Path 1-2-3-6-7
  - Use a stub for A() such that x<=0 is returned

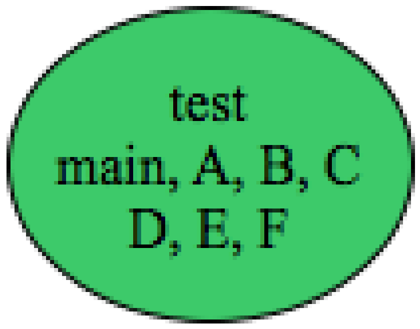
Drivers

- A Driver is a module that calls tested Module(s)
  - Drivers must provide required parameters.
  - Drivers must handle returned values

Big-bang Integration

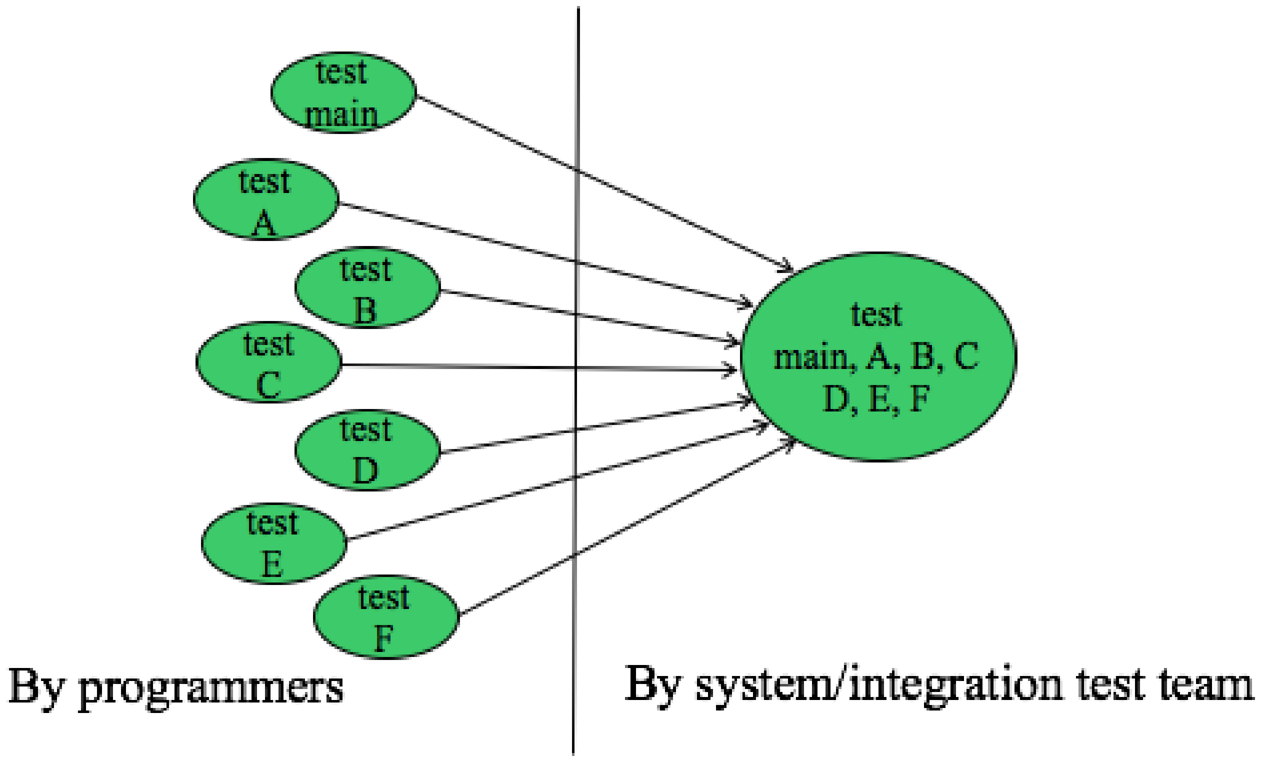
- Non-Incremental Strategy
- Test all components in isolation, then mix them all together and see how it works.
- As all components are merged at once, it is difficult to identify the cause of a failure.





Big Bang Integration

- Assumes all components are initially tested in isolation
- Clear division of responsibilities

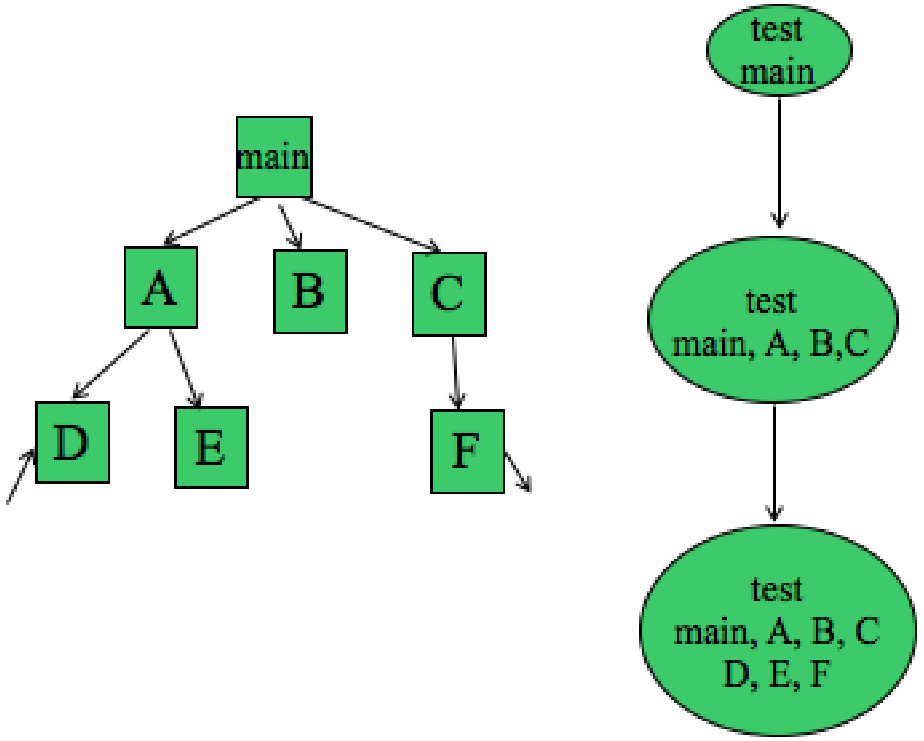


Advantages and Disadvantages

- Advantages
  - Convenient for small systems: Not too difficult to identify/localize errors
  - Can be performed frequently, or continuously: small systems' integration is not computationally expensive
- Disadvantages
  - Fall localization can be difficult for large systems
  - Not very suitable for parallel and incremental development

Top-Down Integration Testing

- Incremental Strategy
- Modules are integrated by moving downward through control hierarchy.
- Modules subordinate to main control module are incorporated
  - Depth-first
  - Breadth-first

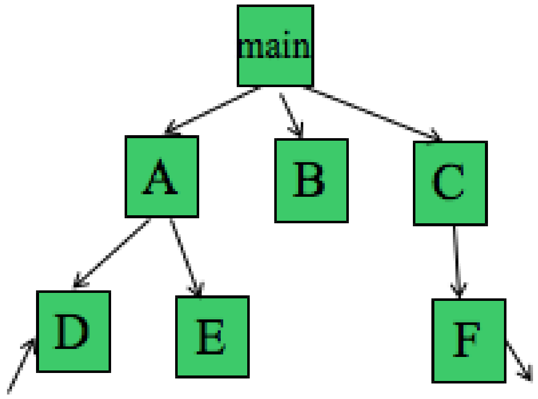


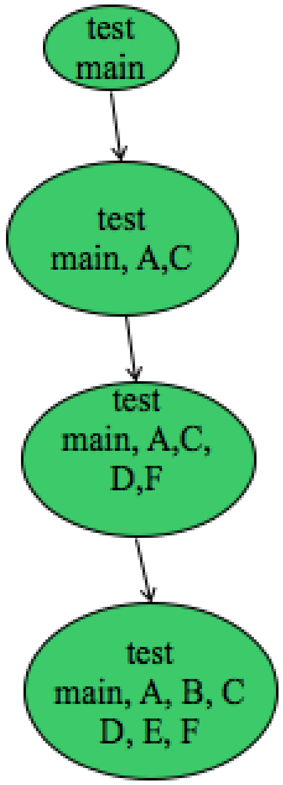
Steps in Top-Down Integration Testing

- The main module is used as a test driver
- Stubs are substituted for all components directly subordinate to the main control module
- Stubs are replaced one at a time with actual components
- Tests are conducted as each component is integrated.

Top-down Integration

- It is possible to alter the order to test as early as possible, for example:
  - To test critical components first
  - To test input/out components first





Evaluation of Top-Down Strategy

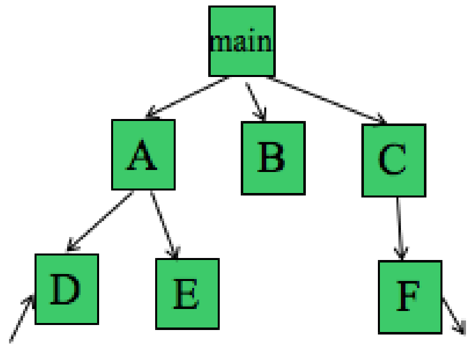
- Verifies major control or decision early in the test process
- Allows early recognition of major problems
- Depth first strategy allows a complete function of the software to be implemented
- Stubs replace lower-level modules so no significant data can flow upward
  - Delay some tests until have actual modules
  - Add code to simulate module

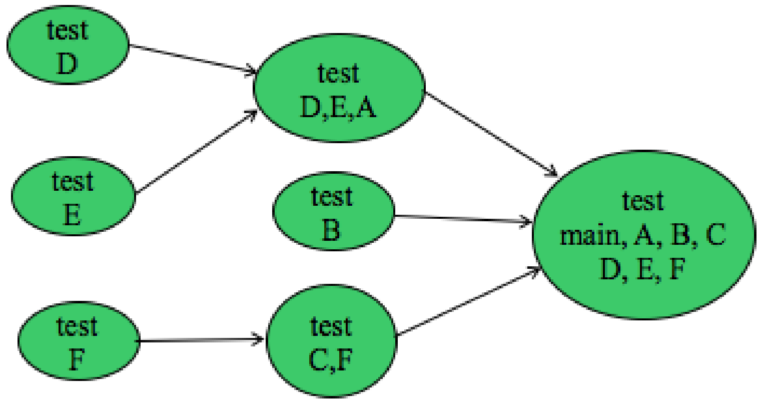
Advantages and Disadvantages

- Advantages:
  - Fault localization is easier
  - Fewer drivers are needed
  - Facilitates testing early prototypes
  - Can accommodate different order of testing and implementation
  - Major design flaws can be identified early because logic/high level design components are usually located at the top of the hierarchy.
- Disadvantages
  - Could require large number of stubs
  - Since reusable components tend to be at the bottom of the hierarchy, they may be inadequately tested.

Bottom-Up Integration

- Low-level components are combined into clusters
- A driver is written to coordinate test case input and output
- Cluster is tested
- Drivers are removed and clusters are combined moving upward in program structure.





Evaluation of Bottom-Up Strategy

- Need for stubs is eliminated (or significantly reduced)
- Operational modules tested thoroughly
- Begins construction and testing with atomic modules

Advantages and Disadvantages

- Advantages
  - Fault localization is simpler than big bang approach
  - No need for stubs
  - Reusable components are tested thoroughly
  - Testing can progress in parallel with implementation
- Disadvantages
  - Need test drivers
  - High level components are tested at the end of the test process (at last and least)
  - Not suitable to build high level skeletal system or prototype for testing

Sandwich Integration

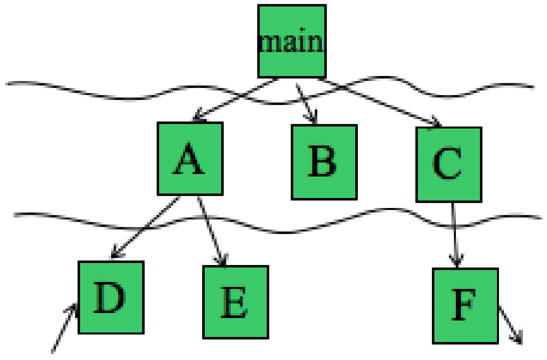
- Combination of bottom-up and top-down integrations
- System is viewed as layers
- Approach 1:
  - Top-down approach is used for the top layer
  - A bottom-up approach is used for the bottom layer
  - Allows integration to begin early in the testing phase
  - Does not test individual components thoroughly before integration

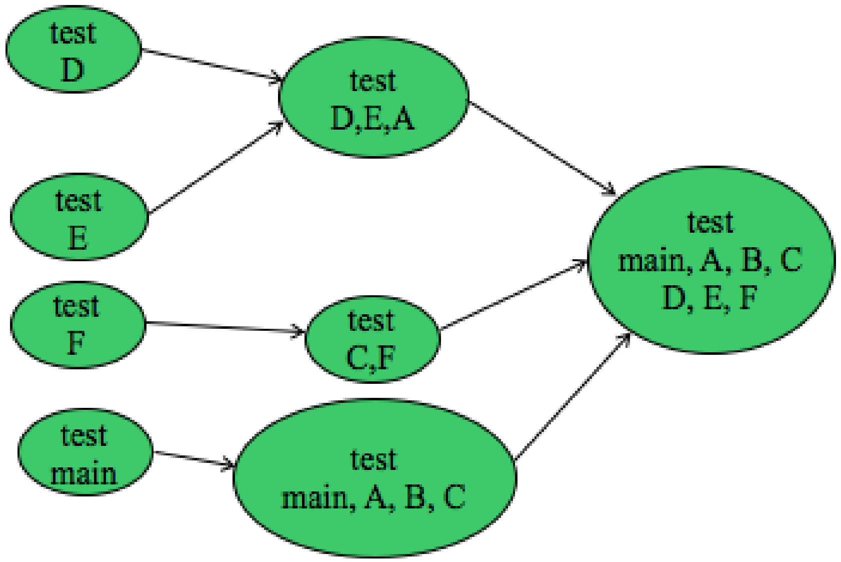
Sandwich integration (Cont.)

- Approach 2:
  - Start with a layer in the middle
  - Use drivers to and stubs to check
  - Work out from middle
  - Allows integration to begin early in the testing phase
  - Does not test individual components thoroughly before integration

Sandwich testing

- Combines top-down and bottom-up
- Three layers
  - Logic (top layer) tested top-down
  - Middle layer
  - Operational (bottom) tested bottom-up





Regression Testing

- Adding new or changing module impacts the system
  - New data flow paths established
  - New I/O may occur
  - New control logic invoked
- Regression testing is re-execution of subset of tests that have already been conducted
  - Ensures changes have not propagated unintended side effects

Regression Test (Cont.)

- Approaches
  - Manual testing
  - Capture/Playback tools: capture test cases and results for subsequent playback and comparison
- Test suite contains following classes of test cases:
  - Representative sample of tests that exercises all software functions
  - Focus on functions likely affected by change
  - Focus on components that have been changed

Risk-Driven Integration Testing

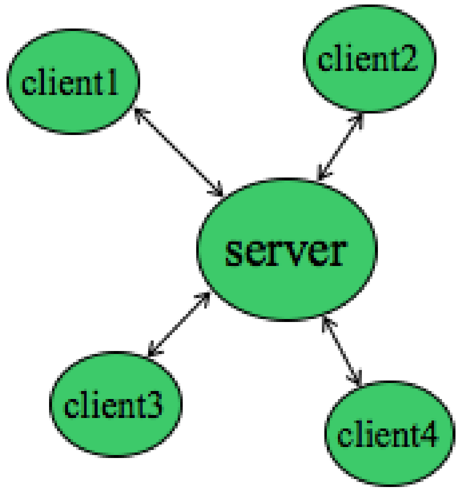
- Integration is based on criticality
- Most critical or complex components are integrated first.
- Facilitates early testing of high-risk components

Object-Oriented Integration using Mock Objects

- Mock Object
  - Designed based on Interfaces
  - Easier to set up and control
  - Isolates code from details that can be filled in later
  - Can be refined incrementally by replacing with actual code
- A **test double** is an object that can stand in for a real object in a test, similar to how a stunt double stands in for an actor in a movie. These are sometimes all commonly referred to as "mocks"

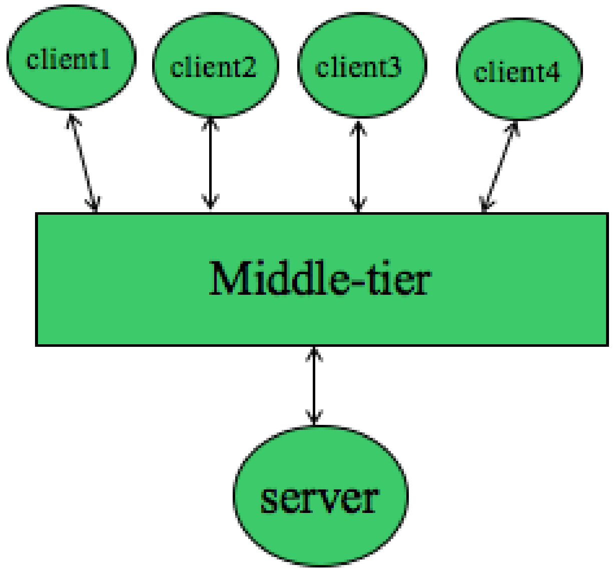
Client Server Integration

- Client-Server systems are two-tier systems
- Test Server with stubs for client types
- Remove all stubs and test server with actual clients
- Same approach work for multi-tier systems



Three-Tier Systems

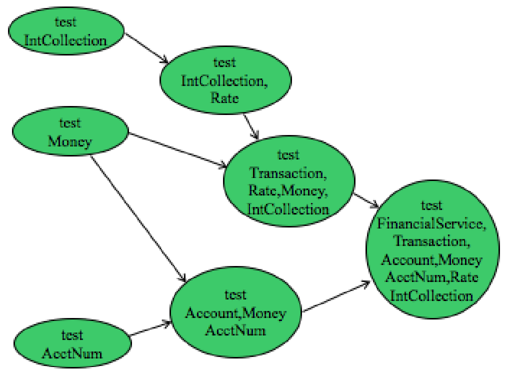
- Test each client with stubs for servers, and the middle-tier.
- Test server with stubs for each client types, and the middle-tier.
- Test each client with middle-tier and server proxy.
- Test server with middle-tier and client proxy.
- Test clients with middle-tier and the actual server.

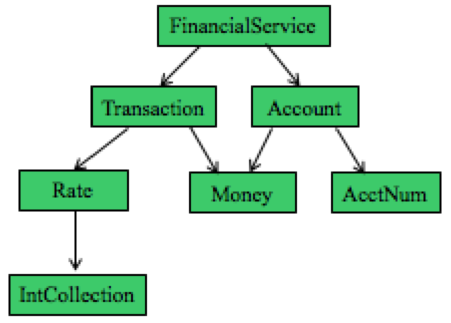


Other types of integration testing

- Function/Thread Integration
  - Integrate components according to threads/functions they belong to.
- Use case based Integration
  - Integrate based on external use cases
- System/Sub-system Integration
  - Integrate sub-systems first
- Cluster Integration
  - Integrate start from leaves units

    and move up in dependency tree


test IntCollection
test IntCollection, Rate
test Money
test Transaction, Rate,Money, IntCollection
test FinancialService, Transaction, Account,Money AcctNum,Rate IntCollection
test Account,Money AcctNum
test AcctNum


FinancialService
Transaction
Account
Rate
Money
AcctNum
IntCollection